

# 基于OpenDigger与MaxKB的 **活动指标时间序列预测&** 仓库活跃度评估

指导老师：王伟

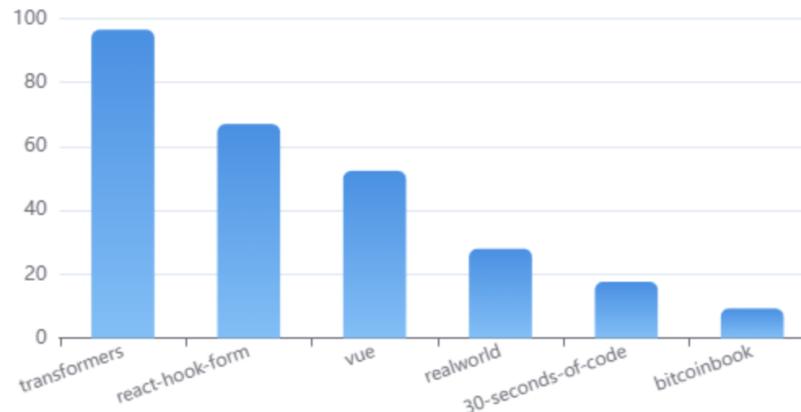
队伍成员：游明东、吴纪红

# 成果速览

Search Repository...

仓库名称模糊搜索

## Random 6 Repos Analysis



仓库活跃分数卡片，活跃状态简评  
*mmdetection*

82.79

仓库整体活跃状态良好，开发节奏稳定，社区互动较为活跃。维护及时，issue/讨论/提交较为频繁。



VS Code 目前的活跃度如何？社区是否依然强劲？

2 条提问

基于 RAG+LangChain  
的大模型问答助手

## 动态数据展示模块



### VS Code 仓库当前发展状况

- 活跃分数: 92.29分
- 活跃状态描述: 仓库活跃度高，开发与社区参与持续活跃，整体运行状态健康。具备良好的持续发展能力。

### 社区及发展状况

- 社区生态: 极其繁荣。扩展市场拥有超 5 万个插件，主题、代码片段、语言包丰富。VS Code 已成为教育、科研、工业开发的标准工具。
- 月活用户: 超过 3000 万，证明其在开发者中的受欢迎程度。
- 更新频率: Microsoft 持续投入大量工程资源，保持每日一次功能更新 (Insiders 版每日更新)。

请输入问题

知识库信息检索，模型分析总结，  
帮助用户快速了解仓库详情。





以下为正式内容

## 2017 年，前端技术浪潮正盛

一名满怀热情的开发者，站在开源社区的入口，试图为自己的前端学习长期投入寻找一个正确的方向。在他的面前，是两个同样活跃、同样耀眼的开源仓库：



REACT



VUE

两个社区讨论热烈、提交频繁、关注者持续增长。但问题在于：

**眼下的“活跃”，是否意味着未来的“可持续”？**

如果未来一年其中一个仓库迅速衰退，

那么他今天的投入，是否会成为一次高风险的选择？

然而，当时的开源生态并没有一个指标，也没有一个针对性的预测方法能够告诉他

**哪个仓库在“未来”仍将保持活跃？**

因此，这带出两个关键问题：

- 1.如何得到准确的仓库未来的活动指标？
- 2.如何基于未来的指标评估仓库的活跃度？

# 1.如何得到准确的仓库未来的活动指标？

本项目考虑基于前沿时序模型预测仓库未来活动指标。所有时间序列预测任务均依赖于高质量的历史数据，因此本项目针对目标仓库的历史数据基于**OpenDigger开源软件生态数据分析挖掘平台**进行系统采集与整理。

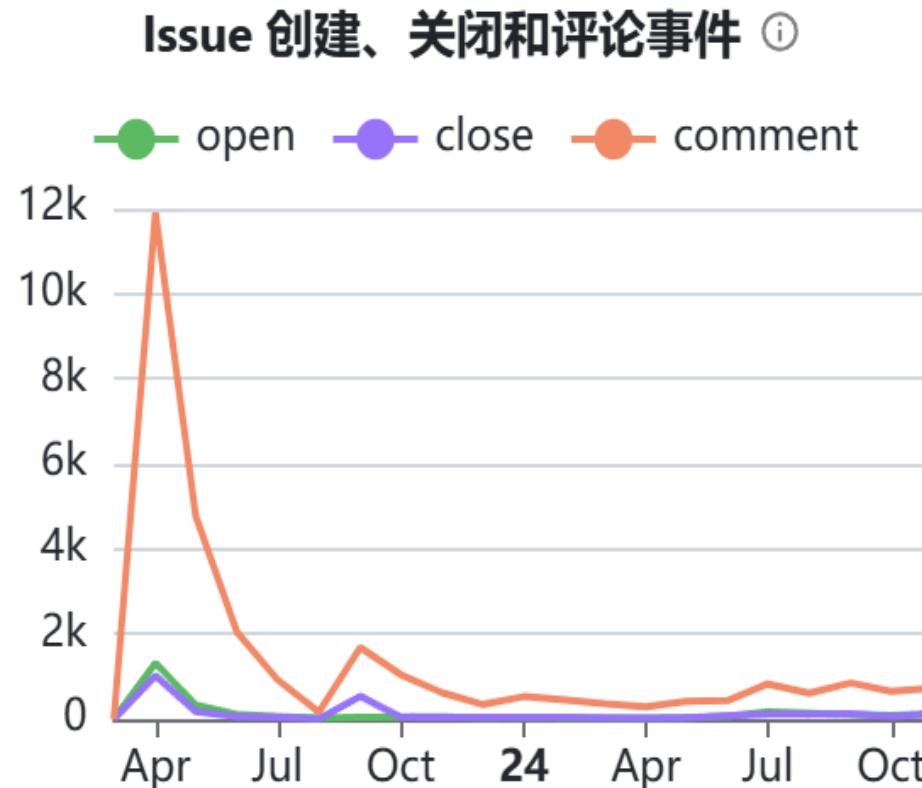
在数据清洗与预处理完成后，形成了覆盖范围广、结构完整的仓库活动时间序列数据集。下，这里从语言类型（Language）、许可证类型（License）以及用户类型（User Type）三个角度进行说明：

- **Language**: 涵盖JavaScript, Python, Java, C, C++, Go等等语言类型，总的语言类型数目超过40，能够涵盖彼时大多数类型的GitHub仓库语言类型。
- **License**: 涵盖MIT, Apache 2.0, BSD 3-Clause, Creative Commons Zero v1.0 Universal等许可证类型，总许可证类别共19种。
- **User\_type**: 本文将user\_type分为两种类型，分别是‘组织’用户和‘个人’用户，在本数据集种，二者的占比分为62.6%，37.4%。

# 开源仓库活动指标常见问题

但是许多仓库的时序走向，缺乏周期性，规律性，人为因素影响大。  
如下图所示。那么如何尽可能的预测准确的仓库活动指标呢？

偏态分布实例: GitHub仓库  
Significant-Gravitas/AutoGPT



人为因素实例: GitHub仓库  
Homebrew/legacy-homebrew



# 开源仓库活动指标预测解决方案

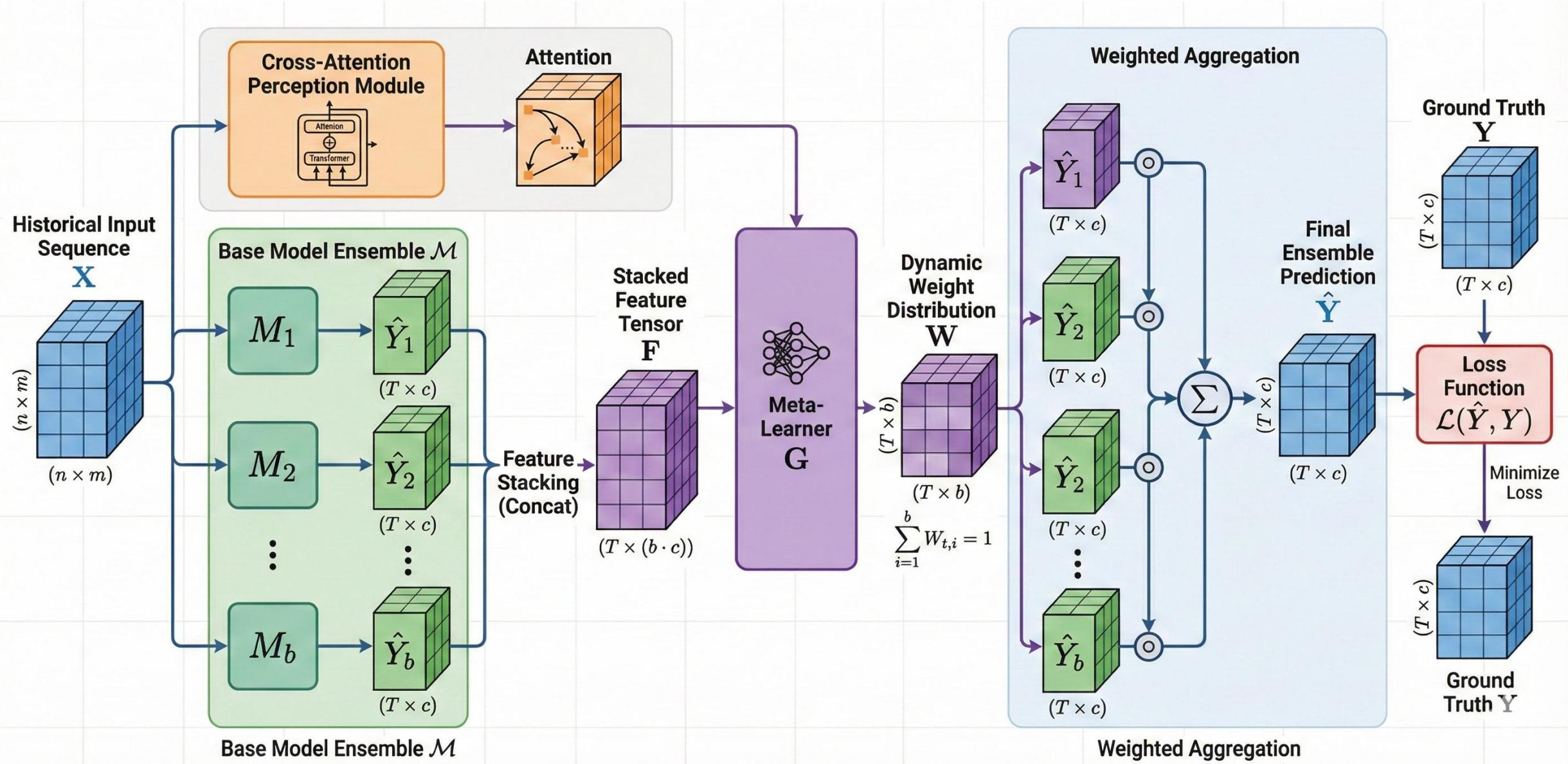


本项目针对开源仓库的数据特性的解决思路是：**不依赖单一模型，而是通过动态集成，让不同模型在各自擅长的场景中发挥作用。**具体而言，整体解决方案包含以下三个关键步骤：

- 1. 多模型协同建模：**不同时间序列模型对数据模式的预测能力存在差异。例如DLinear更适合刻画线性趋势，而 PatchTST 在建模复杂的非线性变化方面表现更优。本文同时引入多种基模型，使其分别捕捉时序数据中的不同结构特征。
- 2. 注意力感知与长期依赖建模：**通过引入注意力感知模块，对历史序列与预测窗口之间的关联关系进行建模，从而提取长期依赖信息，为后续的融合过程提供上下文依据。
- 3. 动态集成与权重自适应：**基于 stacking 集成思想，构建门控网络对各基模型的预测结果进行动态加权。该门控网络能够在预测窗口的不同时间步上，自适应地调整各模型的权重分配。

在上述机制下，仓库的活动指标不再由单一模型给出，而是由**多个“专家模型”共同参与预测**。门控网络综合历史信息与模型输出，在预测窗口的每一步上判断更可信的模型组合，最终得到更加稳定且具有鲁棒性的预测结果。下张内容展示了本文所提框架的工作结构图。

# Context-Aware Dynamic Fusion Ensemble Prediction Framework



# 集成时序预测框架的实验结果

本项目从多个角度对所提出的预测框架进行了系统验证，包括元模型对比、基模型组合对比、消融实验以及参数敏感性分析，以全面评估该框架在策略设计、模型结构和性能表现上的合理性与必要性。

鉴于初赛阶段以方案设计与方法论展示为主，本文未对全部实验结果进行展开说明。后续若有机会进入复赛，期望能够向各位评委进一步完整展示实验细节与研究结论。

表 3.1: 元模型对比试验

元模型	Transformers	React-native	Vscode	Vue	Kubernetes	100Dataset
<b>Attention_fusion</b>	<b>0.2305</b>	0.3501	5.2096	<b>0.1199</b>	<b>0.4145</b>	4.3067
Equal_weighted	0.2592	0.3219	2.1316	0.1312	0.4470	4.3410
Dynamic_fussion	0.3236	0.3251	2.6898	0.1341	0.4284	4.3581
Linear_model	1.2620	2.2990	33.3440	2.9402	4.2166	5.9025
MLP	0.9688	1.8401	31.0273	1.5708	2.4300	5.3933
Weighted_average	0.2492	<b>0.3204</b>	<b>1.3023</b>	0.1209	0.4533	4.4009

表 3.3: 不同模型组合在各数据集上的 MSE 表现及与基模型对比

模型组合	Vue	Transformers	Vscode	React-native	Kubernetes	100Dataset	是否优于组件基模型
PatchTST, DLinear	0.1199	0.2105	5.2096	<b>0.3011</b>	0.3245	4.3067	✓
PatchTST, SegRNN	0.0534	0.2338	<b>1.2956</b>	0.3172	0.3490	4.3962	✓
PatchTST, FrTS	0.0893	0.2541	9.2322	0.3608	0.4642	5.0310	✗
PatchTST, Autoformer	0.0491	0.2247	1.3769	0.3553	0.3757	4.5011	✓
iTransformer, DLinear	0.0718	0.2510	4.0689	0.3912	0.3644	4.8002	✗
iTransformer, TimesNet	0.0454	0.2275	1.5053	0.3573	0.3829	4.8240	✗

表 3.2: 原始模型与 Attention\_aware 增强模型性能对比

模型	Transformers	React-naive	Vscode	Vue	Kubernetes	100Dataset
Autoformer	0.2027	0.3499	1.4133	0.0712	0.3393	5.1028
<b>Autoformer_Atten</b>	<b>0.2004</b>	0.3918	<b>1.3882</b>	0.1099	0.3998	<b>5.0735</b>
DLinear	0.2626	0.4040	4.6576	0.2819	0.4927	4.7324
<b>DLinear_Atten</b>	0.2631	0.4102	5.5118	0.3217	0.5245	<b>4.7100</b>
FrTS	0.2778	0.3631	6.9858	0.1735	0.4306	5.6005
<b>FrTS_Atten</b>	0.3006	<b>0.3604</b>	10.1475	0.2500	0.4562	5.8101
iTransformer	0.2478	0.3210	1.3764	0.0434	0.3978	4.6792
<b>iTransformer_Atten</b>	<b>0.2207</b>	<b>0.3194</b>	<b>1.3285</b>	0.0451	<b>0.3815</b>	4.8575
PatchTST	0.2249	0.3747	1.3308	0.0460	0.3951	4.6773
<b>PatchTST_Atten</b>	0.2477	<b>0.3499</b>	<b>1.2878</b>	0.0498	0.4084	<b>4.5233</b>
SegRNN	0.2078	0.3160	1.5912	0.0577	0.3381	4.8332
<b>SegRNN_Atten</b>	0.2092	0.3169	<b>1.4420</b>	0.0590	<b>0.3339</b>	<b>4.8220</b>
TimesNet	0.2142	0.3609	1.4692	0.0508	0.4023	5.1477
<b>TimesNet_Atten</b>	<b>0.2141</b>	<b>0.3592</b>	<b>1.4518</b>	0.0515	0.4136	<b>5.1240</b>

# 集成时序预测框架的产出与意义



基于上述内容，本项目围绕集成时序预测任务，构建了一套能够有效应对复杂数据范式的预测框架。该框架不仅服务于具体指标的计算，更具备通用的方法论价值，主要体现在以下几个方面：

**更高的预测精度与稳定性：**通过集成多种具有互补优势的时序预测模型，框架能够从不同角度捕捉时间序列中的线性趋势、非线性变化及局部模式，从而缓解单一模型在面对数据异质性和不确定性时性能不稳定的问题。

**更强的模型泛化能力：**在动态融合机制的控制下，最终预测结果能够根据历史上下文信息自适应调整模型权重，使框架能够适配不同类型、不同生命周期阶段的开源仓库，实现统一且鲁棒的预测表现。

**良好的扩展性与易用性：**所提出的框架支持基模型与元模型的模块化插拔设计，提供灵活的超参数配置接口及配套工具函数，能够方便地迁移至其他时间序列预测任务，降低相关研究与实践的使用门槛。

## 2.如何基于未来的指标评估仓库的活跃度？

# 仓库长期活跃状态评估：研究现状



结合一年多以来对相关文献的系统调研，本人得出以下结果：

目前的研究中，**尚未形成**一种针对开源仓库活动指标的、通用且严谨的预测方法。

现有工作更多是从状态判别的角度出发，即往往仅依赖仓库的历史行为数据，将其划分为“活跃”或“非活跃”二分类状态。

这类方法通常依赖人为设定的活跃标准，

然而在开源领域，不同类型仓库在规模、生命周期和社区结构上差异显著，

**缺乏统一且被广泛接受的活跃性定义。**

基于固定阈值的分类范式，在解释性、泛化性以及跨仓库适用性上均存在明显局限。因此本项目跳出分类任务的思路，考虑从仓库未来活动指标出发。

**结合仓库的未来活动指标与数值评估**的方式针对仓库长期活跃度进行评估。

# 活动状态评估指标的确定



仓库活跃度反映的是项目在一段时间内持续吸引参与、促进协作并产出有效贡献的能力。已有研究表明，单一指标（如 Star、Fork）难以刻画真实活跃状态，需从协作流程的多个阶段进行综合衡量。因此，本项目选取如下活动指标：

- **OpenIssue:** 需求与问题提出，反映用户关注度与反馈强度
- **IssueComment:** 问题讨论与互动深度，体现社区协作密度
- **OpenPR:** 代码贡献行为，反映开发活跃度与贡献意愿
- **PRReviewComment:** 代码审查与讨论，体现协作强度与工程规范
- **MergePR:** 有效代码合并，衡量真实产出与治理效率

所选指标覆盖“需求 → 讨论 → 贡献 → 审查 → 合并”的完整开源协作生命周期，相较 Star / Fork 等静态指标，更贴近实际协作过程

仓库活跃度属于综合性评价指标，不存在可直接观测的真实权重。因此，本项目利用开源专家王伟老师等人以及OpenDigger创始人Frank对活动指标的权重作为最终权重。

指标	OpenIssue	IssueComment	OpenPR	PRReviewComment	MergePR
权重	2.22	0.53	4.07	0.74	2.03

## 指标权重解析

**代码贡献为核心（OpenPR / MergePR）：**突出高成本、可落地的开发行为，区分“尝试贡献”与“实际产出”

**需求信号为前置（OpenIssue）：**反映项目被持续使用与关注，但权重低于代码层面行为

**讨论与审查为辅助（IssueComment / PRReviewComment）：**表征协作质量，但不直接产生功能变更，避免被非有效互动放大影响

基于前面的介绍，最终得到如下评估公式：

$$S = \left( \frac{\sum_{i=1}^n W_i g(\bar{x}_i) - 4.7966}{4.7966} \right) \times 100\%.$$

其中， $\bar{x}_i$  第  $i$  个活动指标在未来  $T$  天的活动指标序列  $Y = \{y_t\}_{t=1}^T$  的均值。 $g(\bar{x}_i)$  将  $\bar{x}_i$  经过非线性函数 Sigmoid 进行映射，将结果与指标权重  $W_i$  相乘后进行放缩得到最后的百分制活跃度分数。后文将对其中的公式设计进行详细的展开。

# 仓库活跃度评估公式实验结果

本项目从多个角度以及多个现实世界的金指标对所提出的仓库活跃度评估公式进行验证，包括变体实验、消融实验以及案例分析。

同样鉴于初赛阶段以方案设计与方法论展示为主，本文未对全部实验结果进行展开说明。后续若有机会进入复赛，期望能够向各位评委进一步完整展示实验细节与研究结论。

Model	CHAOSS_Review_Intensity	GT_Intensity_Mean	GT_Stability_InvCV
DTA_Standard (Sigmoid)	<b>0.3378</b>	<b>0.9240</b>	<b>0.8327</b>
DTA_Log	0.3028	0.8975	0.7942
DTA_Tanh	0.2866	0.8972	0.8122

Model	CHAOSS_Review_Intensity	GT_Intensity_Mean	GT_Stability_InvCV
DTA_Standard	<b>0.3378</b>	<b>0.9240</b>	<b>0.8327</b>
DTA_NoWeight	0.2928	0.9018	0.8027
DTA_Linear	0.2969	0.8915	0.7848

评估方式	仓库	transformers	vscode	kubernetes	pandas	react-native	vue	30-seconds-of-code
E_commit		✓	✓	✓	✓	✓	✓	✓
E_release		✓	✓	✓	✗	✓	✓	✗
DTA		96.53	92.29	77.24	65.81	54.48	52.37	17.67
<i>DTA</i> 评分对应预测时间段的未来真实世界活动特征								
GT_Intensity_Mean		4.68	3.96	1.82	2.79	1.15	0.62	0.00
CHAOSS_Review_Intensity		1.01	0.73	1.50	0.60	19.28	0.53	3.54
GT_Stability_InvCV		15.58	8.66	4.27	5.66	3.33	1.97	0.71

# 仓库长期活跃度评估公式的产出与意义



基于上述内容，本项目构建了一套面向长期发展潜力的仓库活跃度评估方法，使评价结果不仅反映历史规模，更能刻画仓库未来的增长趋势与协作潜力。该评估方法的价值主要体现在以下几个方面：

**提升活跃度指标的区分度：**综合 Issue、Pull Request 及其交互行为的未来指标，从活动强度、协作密度与稳定性等维度，更精细地区分不同活跃水平的仓库。

**融合未来趋势，构建可解释评分机制：**基于集成时序预测框架获取未来活动序列，并结合层次分析法（AHP）确定指标权重，在保证可解释性的前提下构建面向未来的活跃度评分。

**实现不同规模仓库的公平比较与潜力评估：**通过数据放缩与非线性边际效应建模，弱化规模惯性对高活跃仓库的影响，同时增强中低活跃区间的变化信号，提升潜在成长仓库的可识别性。

回到最开始的问题，在2017年的时间节点，react 和 vue，谁更适合作为开发者长期学习的对象？下面是本项目中智能问答助手给出的答案。

### 在2017年，react-native仓库活跃状态和vue仓库活跃状态谁更...

随时向我提问吧，让数据帮你洞察开源世界 🚀

在2017年，react-native仓库活跃状态和vue仓库活跃状态谁更好，一个开发者应该选择谁进行长期学习



根据已知信息，我们可以对比2017年时 react-native 和 vue 仓库的活跃状态，并给出一个开发者在当时应该选择哪个进行长期学习的建议。

#### 2017年时的活跃状态

##### React Native

- ◎ 活跃分数：95.71分
- ◎ 活跃状态描述：活跃度约处于历史最佳表现的中间水平，存在一定开发或互动行为，但整体活跃程度有限。

##### Vue

- ◎ 活跃分数：62.79分
- ◎ 活跃状态描述：活跃度约处于历史最佳表现的中间水平，存在一定开发或互动行为，但整体活跃程度有限。



#### 比较与分析

从活跃分数来看，react-native 的活跃分数为95.71分，而 vue 的活跃分数为62.79分。尽管两者都处于“存在开发或互动行为，但整体活跃程度有限”的状态，但 react-native 的活跃度明显更高。



#### 开发者选择建议

在2017年，如果一个开发者需要选择一个框架进行长期学习，可以考虑以下几点：

1. **社区活跃度：** react-native 的活跃度更高，这意味着更多的开发者参与、更频繁的更新和更多的资源支持。
2. **技术栈成熟度：** vue 在2017年已经是一个成熟的框架，核心库稳定且易于上手。而 react-native 正在快速发展，可能会有更多的变化和不稳定性。
3. **应用领域：**如果你主要关注移动应用开发，react-native 是一个更好的选择，因为它专门用于构建跨平台的原生移动应用。而 vue 更适合于Web应用开发。
4. **个人兴趣和项目需求：**最终的选择还应考虑你的个人兴趣和你将要从事的项目类型。如果你对移动应用开发感兴趣，react-native 可能更适合；如果你更倾向于Web应用开发，vue 是一个不错的选择。



#### 结论

在2017年，react-native 的活跃度更高，社区更加活跃。因此，如果你希望进入一个更具活力和技术快速发展的生态系统，react-native 是一个更好的选择。然而，如果你更喜欢一个成熟且稳定的框架，特别是对于Web应用开发，vue 也是一个非常不错的选择。

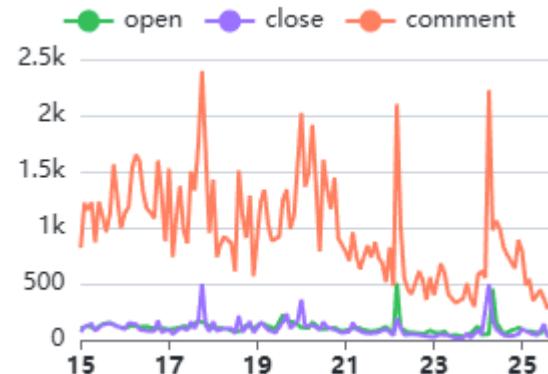
基于上述方法与实验结果得到的活跃分数和模型分析结果，本项目会建议选择 React。

同时从事后发展来看，这一评估结果与真实演化趋势具有较高一致性：在随后的多年中，React 仍持续呈现出较为稳定且周期性的活跃高峰，而 Vue 的整体活跃度则逐渐趋于平缓。该案例表明，本文提出的活跃度评估方法不仅能够刻画仓库在评估时点的状态差异，也在一定程度上反映了其后续发展趋势，从而验证了评分机制在长期活跃度分析中的有效性。

评估分数：95.71



REACT



评估分数：62.79



VUE





# 总结

求实创造 为人师表

**创新性：**在stacking集成策略中**首次**同时引入门控门控和交叉注意力感知模块，实现对多个前沿时序基模型的有效融合，并提出易用性强，可拓展性强的集成时序预测框架。同时提出首个可解释强，区分度高的活跃度评估公式。

**技术挑战：**集成策略对比、融合网络设计、注意力感知模块实现、活跃度评估公式数学合理性分析以及结构设计。以上困难皆被一一克服，同时**公式设计相关成果已发表北大核心期刊**。

**应用场景：**已经通过开源工具 MaxKB 实现智能问答，未来可以考虑通过浏览器插件开发的方式实现如下目标。

- **开源参与决策支持：**帮助贡献者、企业与基金会在资源有限的情况下，识别更具长期发展潜力的项目，优化人力与资金投入。
- **项目健康监测与预警：**辅助仓库维护者及时发现活跃度变化趋势，提前调整项目治理、协作流程或技术路线，提升项目的可持续性。



华东師範大學  
EAST CHINA NORMAL UNIVERSITY

# THANKS

求实创造 为人师表